**Computer Organization 2020**

**HOMEWORK 4 Cache Simulator**

**Due date:**

**Overview**

In memory hierarchy, the small memory, cache, is used to keep data temporarily for increasing the system performance. If the data is in the cache memory, the processor can get the data with high accessing speed. If the data is not in the cache, it is called cache miss, and the processor will read data from main memory. Accessing data from main memory is slower than accessing it from the cache memory. The goal of this homework is to help you understand the cache. In this homework, you need to write a cache simulator. Please follow the specification in this homework and satisfy all the homework requirements.

**General rules for deliverables**

* You need to complete this homework INDIVIDUALLY. You can discuss the homework with other students, but you need to do the homework by yourself. You should not copy anything from someone else, and you should not distribute your homework to someone else. If you violate any of these rules, you will get NEGATIVE scores, or even fail this course directly
* When submitting your homework, compress all files into a single **zip** file, and upload the compressed file to Moodle.
  + Please follow the file hierarchy shown in Figure 1.

**F740XXXXX ( your id ) (folder)**

**src ( folder ) \* Store your source code**

**report.docx ( project report. The report template is already included. Follow the template to complete the report. )**

|  |
| --- |
|  |
| Figure 1. File hierarchy for homework submission |

* **Important!** DO NOT submit your homework in the last minute. Late submission is not accepted.
* You should finish all the requirements (shown below) in this homework and Project report.
* If your code can not be recompiled by TA successfully using Ubuntu 16.04 (x64) gcc-4.4.7, you will receive NO credit.

**Homework Specification**

1. Width of memory (line size) : 32bits
2. Unit of memory : byte.
3. Cache size : Read from test cases. With variable cache size from 1KB up to 256KB.
4. Block size : Read from test cases. From 16B to 256B.
5. Associativity : Read from test cases. Various associativity from direct-mapped to fully associative .
6. Replaced Algorithm : Read from test cases. Will be FIFO or LRU.
7. All cache is initialized 0.

**Input file format**

We will give .txt file which contains the test cases. The file contains a trace of memory accesses executed from some benchmark program.

The 1st line specifies the cache size (kb).

The 2nd line specifies the block size (byte).

The 3rd line specifies the associativity. 0 represents direct-mapped, 1 represents four-way set associative, 2 represents fully associative. The 4th line specifies the Replace algorithm. 0 represents FIFO, 1 represents LRU and 2 represents your policy.

The rest of the test case is a trace of memory accesses executed from some benchmark program.

**trace1.txt**

|  |
| --- |
| **Sample Input** |
| 1024 // cache size (KB)  16 // block size (Byte)  0 // associativity  0 // FIFO=0 , LRU=1, Your Policy=2  0xbfa437cc // No. 1  0xbfa437c8 // No. 2 ……  0xbfa437c4  0xbfa437c0  0xbfa437bc  0xbfa437b8  0xb80437b8  0xb8043794  0xb80437c8  0xb80437cc |

**Output of your cache simulator**

Take trace1 for example, output to file named trace1.out, and output the following information (you need to output which tag is the victim for each request, if there is not any victim then output -1).

|  |
| --- |
| **Sample Output** |
| -1  -1  -1  -1  -1  -1  3066  -1  3066  -1 |

**Homework Requirements**

1. **Report**

Hand in a report answering the following questions

Q1. How do you know the number of block from input file?

Q2. How do you know how many sets in this cache?

Q3. How do you know the bits of the width of the Tag ?

Q4. Briefly describe your data structure of your cache.

Q5. Briefly describe your algorithm of LRU.

Q6. Briefly describe your algorithm of your policy.

Q7. Run trace2.txt, trace3.txt and then makefile to get the miss rate and put it in your report.

1. **Language/Platform**

Please implement this Homework in C or C++ language. The program will be compiled using GCC 4.4.7 on Linux Ubuntu 16.04.

1. **File name**

You need to name your source code as **cache.cpp** or **cache.c**

1. **Command Line Format**

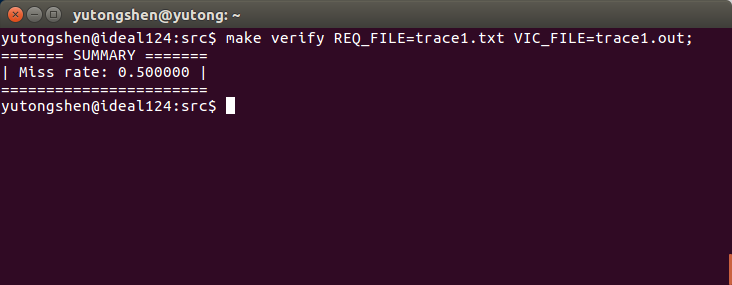
./cache <trace.txt> <trace.out>

Example: ./cache ./trace1.txt ./trace1.out

1. **Verification using makefile**

You can use makefile command to verify your simulation, makefile will read memory request and victim tag information to calculate miss rate

$ make verify REQ\_FILE=<trace.txt> VIC\_FILE=<trace.out>



1. **Note**

Note that you should write your own make file.

|  |
| --- |
| **Important**  When you upload your file, please make sure you have satisfied all the homework requirements, including the **File hierarchy**, **Requirement file** and **Report format**.  If you have any questions, please contact us. |